

[illegible]

☆☆

```

LL               IIIIII             SSSSSSSS
LL               IIIIII             SSSSSSSS
LL               II                 SS
LL               II                 SS
LL               II                 SS
LL               II                 SS
LL               II                 SSSSSS
LL               II                 SSSSSS
LL               II                 SS
LL               II                 SS
LL               II                 SS
LL               II                 SS
LL               II                 SS
LLLLLLLLLLLLLL  IIIIII             SSSSSSSS
LLLLLLLLLLLLLL  IIIIII             SSSSSSSS

```

```
1 0001 0 MODULE OPC$OPCOMMAIN (
2 0002 0 LANGUAGE (BLISS32),
3 0003 0 IDENT = 'V04-000',
4 0004 0 MAIN = OPCOM_MAIN
5 0005 0 ) =
6 0006 0
7 0007 0 *****
8 0008 0 *
9 0009 0 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
10 0010 0 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
11 0011 0 * ALL RIGHTS RESERVED.
12 0012 0 *
13 0013 0 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
14 0014 0 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
15 0015 0 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
16 0016 0 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
17 0017 0 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
18 0018 0 * TRANSFERRED.
19 0019 0 *
20 0020 0 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
21 0021 0 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
22 0022 0 * CORPORATION.
23 0023 0 *
24 0024 0 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
25 0025 0 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
26 0026 0 *
27 0027 0 *
28 0028 0 *****
29 0029 0
30 0030 0 ++
31 0031 0 FACILITY:
32 0032 0
33 0033 0 OPCOM
34 0034 0
35 0035 0 ABSTRACT:
36 0036 0
37 0037 0 This module contains the top leve logic for OPCOM, the
38 0038 0 OperatOr COmunication Manager. OPCOM will provide the
39 0039 0 interface between a user and an operator on the system.
40 0040 0 Specifically, this module contains the routines responsible
41 0041 0 for starting OPCOM in an orderly manner, receiving requests
42 0042 0 and dispatching them to the proper handler, and for the
43 0043 0 orderly shutdown of OPCOM.
44 0044 0
45 0045 0 Environment:
46 0046 0
47 0047 0 VAX/VMS operating system.
48 0048 0
49 0049 0 Author:
50 0050 0
51 0051 0 Steven T. Jeffreys
52 0052 0
53 0053 0 Creation date:
54 0054 0
55 0055 0 March 10, 1981
56 0056 0
57 0057 0 Revision history:
```



```

58      0058 0
59      0059 0
60      0060 0
61      0061 0
62      0062 0
63      0063 0
64      0064 0
65      0065 0
66      0066 0
67      0067 0
68      0068 0
69      0069 0
70      0070 0
71      0071 0
72      0072 0
73      0073 0
74      0074 0
75      0075 0
76      0076 0
77      0077 0
78      0078 0
79      0079 0
80      0080 0
81      0081 0
82      0082 0
83      0083 0
84      0084 0
85      0085 0
86      0086 0
87      0087 0
88      0088 0
89      0089 0
90      0090 0
91      0091 1 BEGIN
92      0092 1
93      0093 1 LIBRARY 'SYS$LIBRARY:LIB.L32';
94      0094 1 LIBRARY 'LIB$:OPCOMLIB';
95      0095 1
96      0096 1

V03-006 CWH3006      CW Hobbs      25-Jul-1984
Remove DEBUG routines to save disk space on kit.

V03-005 CWH3169      CW Hobbs      5-May-1984
Second pass for cluster-wide OPCOM:
- Add RC25, RDRX, TU81 and MAYA microcode messages
- Add conditional assembly code for OPC$INCREASE_STRESS and
  OPC$CHECK_POOL functions

V03-004 CWH3004      CW Hobbs      22-Sep-1983
Make dump conditional on a logical name OPC$DUMP_MAILBOX.

V03-003 CWH3003      CW Hobbs      16-Sep-1983
Add CLUMBX (connection manager) message dispatch.

V03-002 CWH3002      CW Hobbs      30-Jul-1983
Various and sundry things to make OPCOM distributed
across the cluster.

V03-001 RLRV3A1      Robert L. Rappaport 5-Apr-1982
Added MSG$_UDA50MVER and MSG$_DUPUNITNO.

V02-003 STJ0156      Steven T. Jeffreys 05-Feb-1982
Added two new mount verification messages to the
dispatcher table; MVCOMPLETE and MVABORTED.

V02-002 STJ0066      Steven T. Jeffreys 14-Jul-1981
Changed function dispatcher to recognize the mount
verification messages.

--

! Start of OPCOMMAIN

```

```

98      0097 1 GLOBAL ROUTINE OPCOM_MAIN : NOVALUE =
99      0098 1
100     0099 1 ++
101     0100 1 Functional description:
102     0101 1
103     0102 1 This is the main routine for OPCOM. When OPCOM is
104     0103 1 started, control is transferred here. The main routine
105     0104 1 will call a special initialization routine to set things
106     0105 1 up, and then enter its main loop, servicing requests as
107     0106 1 they arrive.
108     0107 1
109     0108 1 Input:
110     0109 1
111     0110 1 None.
112     0111 1
113     0112 1 Implicit Input:
114     0113 1
115     0114 1 None.
116     0115 1
117     0116 1 Output:
118     0117 1
119     0118 1 None.
120     0119 1
121     0120 1 Implicit output:
122     0121 1
123     0122 1 None.
124     0123 1
125     0124 1 Side effects:
126     0125 1
127     0126 1 None.
128     0127 1
129     0128 1 Routine value:
130     0129 1
131     0130 1 None.
132     0131 1 --
133     0132 1
134     0133 2 BEGIN ! Start of OPCOM_MAIN
135     0134 2
136     0135 2 MACRO
137     0136 2
138     0137 2 MSGTYPE = 0,0,16,0%, ! Message type code
139     0138 2 RQSTCODE = 38,0,8,0%, ! Request type code
140     0139 2 CLM_CODE = 39,0,8,0%, ! Request type code for cluster message
141     0140 2
142     0141 2 GLOBAL
143     0142 2
144     0143 2 FINISHED : LONG INITIAL (FALSE); ! Boolean used for loop control
145     0144 2
146     0145 2 EXTERNAL ROUTINE
147     0146 2 DUMP_LOG_FILE, ! Make a formatted dump in the log file
148     0147 2 SHARE_FA0_BUFFER, ! Run something through $FA0
149     0148 2 OPCOM_INIT : NOVALUE, ! OPCOM initialization
150     0149 2 TIME_STAMP : NOVALUE, ! Does periodic timestamp
151     0150 2 WRITE_LOG_FILE, ! Write random string to the log file
152     0151 2
153     0152 2 ! Various message handlers for old format messages.
154     0153 2
```

```
155 0154 2 UNKNOWN_HANDLER : NOVALUE, : Unknown message type handler
156 0155 2 CLUMBX_HANDLER : NOVALUE, : Cluster mailbox messages (special format)
157 0156 2 DEVICE_HANDLER : NOVALUE, : Device online/offline messages (special format)
158 0157 2 TERM_HANDLER : NOVALUE, : Enable operator message handler
159 0158 2 LOG_HANDLER : NOVALUE, : Init logfile message handler
160 0159 2 RQST_HANDLER : NOVALUE, : Request handler
161 0160 2 RPLY_HANDLER : NOVALUE, : Reply handler
162 0161 2 CNCL_HANDLER : NOVALUE, : Cancel handler
163 0162 2 STS_HANDLER : NOVALUE, : Status handler
164 0163 2 SECD_HANDLER : NOVALUE, : Security message handler
165 0164 2
166 0165 2 : The following are message handlers for the new format messages.
167 0166 2
168 0167 2 OPENABLE_HANDLER : NOVALUE, : Operator enable handler
169 0168 2 LOGFILE_HANDLER : NOVALUE, : Logfile control handler
170 0169 2 REQUEST_HANDLER : NOVALUE, : Request handler
171 0170 2 REPLY_HANDLER : NOVALUE, : Reply handler
172 0171 2 CANCEL_HANDLER : NOVALUE, : Cancel handler
173 0172 2 STATUS_HANDLER : NOVALUE, : Status handler
174 0173 2 SHUTDOWN_HANDLER : NOVALUE, : Shutdown handler
175 0174 2 SECURITY_HANDLER : NOVALUE, : Security handler
176 0175 2 ! DEBUG_HANDLER : NOVALUE, : Debug handler, comment out when not needed
177 0176 2 ! CLUSMSG_HANDLER : NOVALUE, : Cluster message handler
178 0177 2
179 0178 2 EXTERNAL
180 0179 2
181 0180 2 GLOBAL_STATUS : BITVECTOR, : Global status bits
182 0181 2 OPER_MBX_CHAN : WORD; : Operator mailbox channel
183 0182 2
184 0183 2 LOCAL
185 0184 2
186 0185 2 RDB : $ref_block, : RDB control structure
187 0186 2 IOSB : $bblock [8], : I/O status block
188 0187 2 REQUEST_BUFFER : $bblock [OPCSK_MAXREAD], : Request receive buffer
189 0188 2 REQUEST_DESC : $desc_block, : Request buffer descriptor
190 0189 2 STATUS : LONG;
191 0190 2
192 0191 2 : Perform the necessary initialization.
193 0192 2
194 0193 2 OPCOM_INIT ();
195 0194 2
196 0195 2
197 0196 2 : Initialize the request buffer descriptor.
198 0197 2
199 0198 2
200 0199 2 REQUEST_DESC [DSC$B_DTYPE] = 0;
201 0200 2 REQUEST_DESC [DSC$B_CLASS] = 0;
202 0201 2 REQUEST_DESC [DSC$A_POINTER] = REQUEST_BUFFER;
203 0202 2
204 0203 2 : Enter the main loop.
205 0204 2
206 0205 2
207 0206 2 WHILE NOT .FINISHED DO
208 0207 2 BEGIN
209 0208 2 : If a timestamp is pending and OPCOM is not busy with
210 0209 2 : a request, then do the timestamp.
211 0210 2
```



```
212 0211 !
213 0212 GLOBAL STATUS [GBLSTS_K_BUSY] = FALSE; ! OPCOM is not busy
214 0213 IF .GLOBAL_STATUS [GBLSTS_K_TIMESTAMP_PENDING]
215 0214 THEN
216 0215     TIME_STAMP ();
217 0216
218 0217 !
219 0218 ! Issue a read request to the operator mailbox.
220 0219
221 0220 IF NOT (STATUS = $QIOW(FUNC = IOS$ READVBLK,
222 0221     CHAN = .OPER_MBX_CHAN,
223 0222     IOSB = IOSB,
224 0223     EFN = EFN_K_MAILBOX,
225 0224     P1 = REQUEST_BUFFER,
226 0225     P2 = OPC$K_MAXREAD
227 0226     ))
228 0227 THEN
229 0228     $signal_stop (.STATUS);
230 0229
231 0230 !
232 0231 ! Check the status of the read.
233 0232
234 0233 IF NOT (STATUS = .IOSB [0,0,16,0])
235 0234 THEN
236 0235     IF .STATUS NEQ SS$_ENDOFFILE ! A COPY to _MBA2: will produce this
237 0236     THEN
238 0237         $signal_stop (.STATUS);
239 0238
240 0239 !
241 0240 ! Since OPCOM now has a request to service, set the GBLSTS_K_BUSY bit.
242 0241 ! This serves as an interlock to prevent the asynchronous
243 0242 ! timestamp function from going off at an inappropriate time.
244 0243
245 0244 GLOBAL_STATUS [GBLSTS_K_BUSY] = TRUE;
246 0245
247 0246 !
248 0247 ! Set the request buffer length in the descriptor.
249 0248
250 0249 REQUEST_DESC [DSC$W_LENGTH] = .IOSB [2,0,16,0];
251 0250
252 0251 !
253 0252 ! For debugging, write the message into the log file, and flush
254 0253
255 0254 BEGIN
256 0255 LOCAL
257 0256     lcl_buf : $bvector [16],
258 0257     out_dsc : VECTOR [2, LONG];
259 0258     out_dsc [0] = 16;
260 0259     out_dsc [1] = lcl_buf;
261 0260 IF $trnlog (lognam=%ASCII 'OPC$DUMP_MAILBOX', rsllen=out_dsc, rslbuf=out_dsc) EQL ss$_normal
262 0261 THEN
263 0262     DUMP_LOG_FILE (REQUEST_DESC, %ASCII 'Record received in mailbox');
264 0263
265 0264 END;
266 0265
267 0266 !
268 0267 ! For debugging, check the cluster configuration at every mailbox read. To put additional
! stress on the system, request acks from every node if the configuration has changed. Make
```

```
! all this noise switchable by defining the logical name OPC$INCREASE_STRESS (presence of
! the name is significant, not its value).
! Only include if /VARIANT specified
%IF %VARIANT GTR 0
%THEN
BEGIN
LOCAL          lcl_buf : $bvector [16], out_dsc : VECTOR [2, LONG];
EXTERNAL       lcl_csid, nod_head : vector [2, long];
EXTERNAL ROUTINE clusutil_configure, clusmsg_ack_please;
out_dsc [0] = 16; out_dsc [1] = lcl_buf;
IF $strnlog (lognam=%ASCII 'OPC$INCREASE_STRESS', rslten=out_dsc, rslbuf=out_dsc) EQL ss$_normal
THEN
BEGIN
IF clusutil_configure ()
THEN
BEGIN
LOCAL          nod : $ref_bblock;
nod = .nod_head [0];
WHILE .nod NEQ nod_head [0]
DO
BEGIN
IF .nod [nod_l_node_csid] NEQ .lcl_csid
THEN
BEGIN
nod [nod_v_ack_pend] = false;
CLUSMSG_ACK_PLEASE (.nod);
END;
nod = .nod [nod_l_flink];
END;
END;
END;
%FI
! End of conditionally compiled message codes.

! For debugging, check pool for corruption. N.B. this puts a severe load on the system at IPL 11!
! Only include if /VARIANT specified
%IF %VARIANT GTR 0
%THEN
BEGIN
LOCAL          lcl_buf : $bvector [16], out_dsc : VECTOR [2, LONG];
out_dsc [0] = 16; out_dsc [1] = lcl_buf;
IF $strnlog (lognam=%ASCII 'OPC$CHECK_POOL', rslten=out_dsc, rslbuf=out_dsc) EQL ss$_normal
THEN
BEGIN
LOCAL
arglist : VECTOR [2, LONG];
EXTERNAL ROUTINE
monitor_pool;
arglist [0] = 1;
arglist [1] = %X'1111';
$CMKRNL (routin=monitor_pool, arglst=arglist);
END;
END;
%FI
! End of conditionally compiled debugging code.

!
```


Dispatch the request to the proper handler. Some messages do not come through \$SNDOPR, and require special treatment. For example, device on/offline messages are sent via a call to EX\$SNDEVMSG, and are in a different format from most of the other known message types.

SELECTONEU .REQUEST_BUFFER [MSGTYPE] OF

```
SET
[MSG$_CLUMBX] : CLUMBX_HANDLER (REQUEST_DESC);
[MSG$_DEVOFFLIN] : DEVICE_HANDLER (REQUEST_DESC);
[MSG$_DEVONLIN] : DEVICE_HANDLER (REQUEST_DESC);
[MSG$_DEVOFFLINX] : DEVICE_HANDLER (REQUEST_DESC);
[MSG$_WRONGVOL] : DEVICE_HANDLER (REQUEST_DESC);
[MSG$_DEVWRTLCK] : DEVICE_HANDLER (REQUEST_DESC);
[MSG$_MVCOMPLETE] : DEVICE_HANDLER (REQUEST_DESC);
[MSG$_MVABORTED] : DEVICE_HANDLER (REQUEST_DESC);
[MSG$_UDA50MVER] : DEVICE_HANDLER (REQUEST_DESC);
[MSG$_DUPUNITNO] : DEVICE_HANDLER (REQUEST_DESC);
[MSG$_TM78MVER] : DEVICE_HANDLER (REQUEST_DESC);
[MSG$_RC25MVER] : DEVICE_HANDLER (REQUEST_DESC);
[MSG$_RDRXMVER] : DEVICE_HANDLER (REQUEST_DESC);
[MSG$_TU81MVER] : DEVICE_HANDLER (REQUEST_DESC);
[MSG$_MAYAMVER] : DEVICE_HANDLER (REQUEST_DESC);
```

[MSG\$_OPRQST] : BEGIN

Dispatch the request to the proper handler.

CASE .REQUEST_BUFFER [RQSTCODE]

FROM 0 TO OPCS_X_REQUEST_END_MARK-1 OF
SET

The following request types are in the old format.

```
[OPCS_RQ_TERME] : TERME_HANDLER (REQUEST_DESC);
[OPCS_RQ_LOGI] : LOGI_HANDLER (REQUEST_DESC);
[OPCS_RQ_RQST] : RQST_HANDLER (REQUEST_DESC);
[OPCS_RQ_REPLY] : RPLY_HANDLER (REQUEST_DESC);
[OPCS_RQ_CANCEL] : CNCL_HANDLER (REQUEST_DESC);
[OPCS_RQ_STATUS] : STS_HANDLER (REQUEST_DESC);
[OPCS_RQ_SECURITY] : SECD_HANDLER (REQUEST_DESC);
```

The following request types are in the new format.

```
[OPCS_X_OPRENABLE] : OPRENABLE_HANDLER (REQUEST_DESC);
[OPCS_X_LOGFILE] : LOGFILE_HANDLER (REQUEST_DESC);
[OPCS_X_REQUEST] : REQUEST_HANDLER (REQUEST_DESC);
[OPCS_X_REPLY] : REPLY_HANDLER (REQUEST_DESC);
[OPCS_X_CANCEL] : CANCEL_HANDLER (REQUEST_DESC);
[OPCS_X_STATUS] : STATUS_HANDLER (REQUEST_DESC);
[OPCS_X_SHUTDOWN] : SHUTDOWN_HANDLER (REQUEST_DESC);
[OPCS_X_SECURITY] : SECURITY_HANDLER (REQUEST_DESC);
```

Debug handler, comment out when not necessary

! [OPCS_X_DEBUG] : DEBUG_HANDLER (REQUEST_DESC);

Requests for cluster-related messages.

```
! End of OPCOM_MAIN
```

```

.TITLE OPC$OPCOMMAIN
.IDENT \V04-000\

.PSECT $SPLITS$,NOWRT,NOEXE,2

4F 42 4C 49 41 4D 5F 50 4D 55 44 24 43 50 4F 00000 P.AAB: .ASCII \OPC$DUMP_MAILBOX\
                                     58 0000F
                                     010E0010 00010 P.AAA: .LONG 17694736
                                     000000000' 00014 .ADDRESS P.AAB
64 65 76 69 65 63 65 72 20 64 72 6F 63 65 52 00018 P.AAD: .ASCII \Record received in mailbox\<0><0>
    00 00 78 6F 62 6C 69 61 6D 20 6E 69 20 00027
                                     010E001A 00034 P.AAC: .LONG 17694746
                                     000000000' 00038 .ADDRESS P.AAD

.PSECT $GLOBAL$,NOEXE,2

00000000 00000 FINISHED::
                                     .LONG 0

.EXTRN DUMP_LOG_FILE, SHARE_FAO_BUFFER
.EXTRN OPCOM_INIT, TIME_STAMP
.EXTRN WRITE_LOG_FILE, UNKNOWN_HANDLER
.EXTRN CLUMBX_HANDLER, DEVICE_HANDLER
.EXTRN TERME_HANDLER, LOGI_HANDLER
.EXTRN RQST_HANDLER, RPLY_HANDLER
.EXTRN CNCL_HANDLER, STS_HANDLER
.EXTRN SECU_HANDLER, OPRENABLE_HANDLER
.EXTRN LOGFILE_HANDLER
.EXTRN REQUEST_HANDLER
.EXTRN REPLY_HANDLER, CANCEL_HANDLER
.EXTRN STATUS_HANDLER, SHUTDOWN_HANDLER
.EXTRN SECURITY_HANDLER
.EXTRN CLUSMSG_HANDLER
.EXTRN GLOBAL_STATUS, OPER_MBX_CHAN
.EXTRN SYSSQIDW, LIB$STOP
.EXTRN SYS$TRNLOG

.PSECT $CODE$,NOWRT,2

```


			001C	00000	.ENTRY	OPCOM MAIN, Save R2,R3,R4	: 0097
	54	0000G	CF	9E	MOVAB	GLOBAL STATUS, R4	:
	5E	F5D8	CE	9E	MOVAB	-2600(SP), SP	:
	CF		00	FB	CALLS	#0, OPCOM_INIT	: 0194
		1A	AE	B4	CLRW	REQUEST_DESC+2	: 0199
	1C	20	AE	9E	MOVAB	REQUEST_BUFFER, REQUEST_DESC+4	: 0201
	01	0000	CF	E9	BLBC	FINISHED, 2\$	: 0206
				04	RET		:
	64	40	8F	8A	BICB2	#64, GLOBAL STATUS	: 0212
05	64		05	E1	BBC	#5, GLOBAL STATUS, 3\$	: 0213
	0000G		00	FB	CALLS	#0, TIME_STAMP	: 0215
			7E	7C	CLRW	-(SP)	: 0226
			7E	7C	CLRW	-(SP)	:
	7E	0A00	8F	3C	MOVZWL	#2560, -(SP)	:
		34	AE	9F	PUSHAB	REQUEST_BUFFER	:
			7E	7C	CLRW	-(SP)	:
		F8	AD	9F	PUSHAB	IOSB	:
			31	DD	PUSHL	#49	:
	7E	0000G	CF	3C	MOVZWL	OPER_MBX_CHAN, -(SP)	:
			03	DD	PUSHL	#3	:
	00000000G	00	0C	FB	CALLS	#12, SYSSQIOW	:
		53	50	D0	MOVL	R0, STATUS	:
		10	53	E9	BLBC	STATUS, 4\$	:
		53	F8	AD	MOVZWL	IOSB, STATUS	: 0233
		13	53	E8	BLBS	STATUS, 5\$	:
	00000870	8F	53	D1	CMPL	STATUS, #2160	: 0235
			0A	13	BEQL	5\$	:
			53	DD	PUSHL	STATUS	: 0237
	00000000G	00	01	FB	CALLS	#1, LIB\$STOP	:
				04	RET		:
		40	8F	88	BISB2	#64, GLOBAL STATUS	: 0244
	18	FA	AD	B0	MOVW	IOSB+2, REQUEST_DESC	: 0249
			10	D0	MOVL	#16, OUT_DSC	: 0258
	04	08	AE	9E	MOVAB	LCL BUF, -OUT_DSC+4	: 0259
			7E	7C	CLRW	-(SP)	: 0260
			7E	D4	CLRL	-(SP)	:
		0C	AE	9F	PUSHAB	OUT_DSC	:
		10	AE	9F	PUSHAB	OUT_DSC	:
		0000	CF	9F	PUSHAB	P.AXA	:
	00000000G	00	06	FB	CALLS	#6, SYS\$TRNLOG	:
		01	50	D1	CMPL	R0, #1	:
			0C	12	BNEQ	6\$	:
		0000	CF	9F	PUSHAB	P.AAC	: 0262
		1C	AE	9F	PUSHAB	REQUEST_DESC	:
	0000G	CF	02	FB	CALLS	#2, DUMP LOG FILE	:
		52	AE	3C	MOVZWL	REQUEST_BUFFER, R2	: 0329
	0059	8F	52	B1	CMPL	R2, #89	: 0331
			0A	12	BNEQ	7\$	:
		18	AE	9F	PUSHAB	REQUEST_DESC	:
	0000G	CF	01	FB	CALLS	#1, CLUMBX_HANDLER	:
			66	11	BRB	9\$	:
		05	52	B1	CMPL	R2, #5	: 0332
			59	13	BEQL	8\$	:
		07	52	B1	CMPL	R2, #7	: 0333
			54	13	BEQL	8\$	:
	0050	8F	52	B1	CMPL	R2, #80	: 0334

			4D	13	000C8	BEQL	8\$				
0051	8F		52	B1	000CA	CMPW	R2, #81			0335	
			46	13	000CF	BEQL	8\$				
0052	8F		52	B1	000D1	CMPW	R2, #82			0336	
			3F	13	000D6	BEQL	8\$				
0054	8F		52	B1	000D8	CMPW	R2, #84			0337	
			38	13	000DD	BEQL	8\$				
0055	8F		52	B1	000DF	CMPW	R2, #85			0338	
			31	13	000E4	BEQL	8\$				
0057	8F		52	B1	000E6	CMPW	R2, #87			0339	
			2A	13	000EB	BEQL	8\$				
0058	8F		52	B1	000ED	CMPW	R2, #88			0340	
			23	13	000F2	BEQL	8\$				
005A	8F		52	B1	000F4	CMPW	R2, #90			0341	
			1C	13	000F9	BEQL	8\$				
005D	8F		52	B1	000FB	CMPW	R2, #93			0342	
			15	13	00100	BEQL	8\$				
005E	8F		52	B1	00102	CMPW	R2, #94			0343	
			0E	13	00107	BEQL	8\$				
005F	8F		52	B1	00109	CMPW	R2, #95			0344	
			07	13	0010E	BEQL	8\$				
0060	8F		52	B1	00110	CMPW	R2, #96			0345	
			0A	12	00115	BNEQ	10\$				
		18	AE	9F	00117	PUSHAB	REQUEST_DESC				
0000G	CF		01	FB	0011A	CALLS	#1, DEVICE_HANDLER				
			7D	11	0011F	BRB	20\$				
	08		52	B1	00121	CMPW	R2, #8			0347	
			31	12	00124	BNEQ	12\$				
		15	AE	8F	00126	CASEB	REQUEST_BUFFER+38, #0, #21			0351	
0043		0039	00CF		0012B	.WORD	34\$-11\$,-				
006B		0061	004D		00133		13\$-11\$,-				
007F		0075	00CF		0013B		14\$-11\$,-				
00A7		009D	0089		00143		15\$-11\$,-				
00C5		00BB	00B1		0014B		16\$-11\$,-				
			00CF		00153		17\$-11\$,-				
							18\$-11\$,-				
							19\$-11\$,-				
							34\$-11\$,-				
							34\$-11\$,-				
							21\$-11\$,-				
							22\$-11\$,-				
							23\$-11\$,-				
							24\$-11\$,-				
							25\$-11\$,-				
							26\$-11\$,-				
							28\$-11\$,-				
							34\$-11\$,-				
							30\$-11\$,-				
							32\$-11\$,-				
							34\$-11\$,-				
							34\$-11\$				
			00A0	31	00157	BRW	34\$			0386	
			18	AE	9F	PUSHAB	REQUEST_DESC			0357	
0000G	CF		01	FB	0015D	CALLS	#1, TERM_HANDLER				
			76	11	00162	BRB	27\$				
			18	AE	9F	PUSHAB	REQUEST_DESC			0358	
0000G	CF		01	FB	00167	CALLS	#1, LOG_HANDLER				

		18	76	11	0016C	BRB	29\$		
0000G	CF		AE	9F	0016E	PUSHAB	REQUEST_DESC	0359	
			01	FB	00171	CALLS	#1, RQST_HANDLER		
		18	76	11	00176	BRB	31\$		
0000G	CF		AE	9F	00178	PUSHAB	REQUEST_DESC	0360	
			01	FB	0017B	CALLS	#1, RPLY_HANDLER		
		18	76	11	00180	BRB	33\$		
0000G	CF		AE	9F	00182	PUSHAB	REQUEST_DESC	0361	
			01	FB	00185	CALLS	#1, CNCL_HANDLER		
		18	76	11	0018A	BRB	35\$		
0000G	CF		AE	9F	0018C	PUSHAB	REQUEST_DESC	0362	
			01	FB	0018F	CALLS	#1, STS_HANDLER		
		18	6C	11	00194	BRB	35\$		
0000G	CF		AE	9F	00196	PUSHAB	REQUEST_DESC	0363	
			01	FB	00199	CALLS	#1, SECD_HANDLER		
		18	62	11	0019E	BRB	35\$		
0000G	CF		AE	9F	001A0	PUSHAB	REQUEST_DESC	0367	
			01	FB	001A3	CALLS	#1, OPRENABLE_HANDLER		
		18	58	11	001A8	BRB	35\$		
0000G	CF		AE	9F	001AA	PUSHAB	REQUEST_DESC	0368	
			01	FB	001AD	CALLS	#1, LOGFILE_HANDLER		
		18	4E	11	001B2	BRB	35\$		
0000G	CF		AE	9F	001B4	PUSHAB	REQUEST_DESC	0369	
			01	FB	001B7	CALLS	#1, REQUEST_HANDLER		
		18	44	11	001BC	BRB	35\$		
0000G	CF		AE	9F	001BE	PUSHAB	REQUEST_DESC	0370	
			01	FB	001C1	CALLS	#1, REPLY_HANDLER		
		18	3A	11	001C6	BRB	35\$		
0000G	CF		AE	9F	001C8	PUSHAB	REQUEST_DESC	0371	
			01	FB	001CB	CALLS	#1, CANCEL_HANDLER		
		18	30	11	001D0	BRB	35\$		
0000G	CF		AE	9F	001D2	PUSHAB	REQUEST_DESC	0372	
			01	FB	001D5	CALLS	#1, STATUS_HANDLER		
		18	26	11	001DA	BRB	35\$		
0000G	CF		AE	9F	001DC	PUSHAB	REQUEST_DESC	0373	
			01	FB	001DF	CALLS	#1, SHUTDOWN_HANDLER		
		18	1C	11	001E4	BRB	35\$		
0000G	CF		AE	9F	001E6	PUSHAB	REQUEST_DESC	0374	
			01	FB	001E9	CALLS	#1, SECURITY_HANDLER		
		18	12	11	001EE	BRB	35\$		
0000G	CF		AE	9F	001F0	PUSHAB	REQUEST_DESC	0382	
			01	FB	001F3	CALLS	#1, CLUSMSG_HANDLER		
		18	08	11	001F8	BRB	35\$		
0000G	CF		AE	9F	001FA	PUSHAB	REQUEST_DESC	0329	
			01	FB	001FD	CALLS	#1, UNKNOWN_HANDLER	0394	
			FE14	31	00202	BRW	1\$	0206	
			04	00205	RET			0398	

; Routine Size: 518 bytes, Routine Base: \$CODE\$ + 0000

: 400 0399 1
: 401 0400 1 END
: 402 0401 0 ELUDOM

! End of OPCOMMAIN

PSECT SUMMARY

Name	Bytes	Attributes
\$GLOBALS	4	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	60	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	518	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	36	0	1000	00:01.8
\$255\$DUA28:[OPCOM.OBJ]OPCOMLIB.L32;1	633	20	3	43	00:00.9

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:OPCOMMAIN/OBJ=OBJ\$:OPCOMMAIN MSRC\$:OPCOMMAIN/UPDATE=(ENH\$:OPCOMMAIN)

: Size: 518 code + 64 data bytes
: Run Time: 00:12.6
: Elapsed Time: 00:35.0
: Lines/CPU Min: 1906
: Lexemes/CPU-Min: 11576
: Memory Used: 180 pages
: Compilation Complete

0290 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

